NORTH /AM AMATEUR TS PROGRAMMERS' N.L.

William J. Harmer
97 Ruskin Street
Ottawa, Ont. K1Y 4B3

New Form - New Emphasis T/S, QL, Amstrad, Cambridge Interested.

Volume#3N ---Issue#1-©1992 in the public domain, photocopy freely ---- 12 pages

It has been almost a year since the last issue of the "Northern Computer Hobby User" newsletter came out and it is starting to be a matter of regret that it was shut down. So, the idea occurred quite late one night, to revive it, and change the name from simply 'Northern' which may have seemed to mean different things to different folks who might ask, "Northern Canada?"(1300 miles from here to the Northwest Territories), "North of the Canada-US border?" or "North of the Mason-Dixon Line?" to North/Am, for North America. Perhaps we can find some Spanish computer publications to review to really get into the spirit! Also, the focus of the content is being changed from trying to predict the future of home computers, etc., which seems to be almost completely unpredictable right now, this being now a major turning point in the history of microcomputers. We shall highlight things that we know best like our favourite Timex-Sinclair computers that use the Z-80 microprocessor but go one further. When the local Timex-Sinclair user group was still functioning, this writer moved that Amstrad computers be admitted to the fold, and the motion was turned down and everyone glared at me, like my shoe had tracked something into the room with a most unpleasant odour. Well, the club having departed, (with only a few whimpers) there is no one left to out-vote me, and Amstrad computers, Z-80, CP/M, MS DOS, DOS Plus, whatever, are being added to the fold, moved, passed and carried with a unaminous vote of one. My other alternative suggestion to turn the club into an amateur programmer's club, will also be tested in the pages of this newsletter, by covering the art, craft and hobby of sitting down at your computer with BASIC up and running and trying to come up with a program that has never run anywhere before you tried it, namely, something original. That for me is my reason for the hobby and any groups.

What this newsletter is not, is narrow, for commercial or professional use, or designed to replace any other newsletter than the before-mentioned Northern Computer Hobby User N/L of a few years ago, copies of which may not all be still available, for those of you interested in them. For a few dollars perhaps a couple of old issues of that could be dredged up still, but this newsletter is trying to make a new start completely. Read on and enjoy, and make a few photocopies of this issue for your friends if you want to. For the time being, whether subscriptions will be able to be offered, is still a moot point. First, we shall see how the first issue is received, in other words, this one you hold in your hands this moment.

Featuring: Thoughts On Office Automation And Where It Is Leading Computers (p.5)
CP/M, Disk System Progress, Book Reviews, Some Reprinted Articles
This Is A Test Issue. Mission Statement: To Enjoy!

Local News: With the fall, people think of indoor activities, like, well, attending user group meetings, and in the Ottawa area there are many active groups, although Timex-Sinclair group folded two years ago, when officers felt that they had little patience and time to devote to it, or at least not enough to keep it going in a time of declining membership. The fact that newsletters produced by this publisher were never financed by the local group, even though they were Timex-Sinclair in subject matter, means only that the copies once distributed free at local meetings cannot be distributed that way and are too expensive to mail. If you are a former TS user group member interested in receving this newsletter, give me a shout anyway. TS user groups out-of-town are still thriving, in Toronto, Vancouver and a number of cities in the USA. So there is no excuse for not being a user group member, even if only of an out of town group. In fact the newsletter of the Toronto group is thicker than ever. They may be contacted via Mr.George Chambers, 14 Richome Ct., Scarborough, Ont., Canda, M1K 2Y1 or failing that via telephone (Jeff Taylor, 416-244-8583). Some day we should publish a list of user groups known to us. Give us a letter if you think your user group may not be getting noticed enough so we can mention it here.

This newsletter: This is a test issue and may be followed by a few more or maybe not. No subscriptions are anticipated to be offered. Copies will merely be sent to (very) few user groups to photocopy and distribute if they choose. That is the beginning and end of my obligation. Back issues will not be kept. To make sure of a reply to correspondence, please include a couple of bucks towards the expense of mailing a reply and enclosures that this writer might wish to send. Photocopies are not cheap, envelopes cost money and shoe leather to get to the copy shops even must be taken into account in the end (isn't it nice to know accounting?).

Machine Code And How To Tailor It For Speed, And Why (68000)

Everyone has heard that machine code is faster than BASIC, especially ROM'd BASIC. The surprising thing is that it may not in some cases, for various reasons but it may be the the overhead in calling a machine code subroutine from BASIC that makes it slow. This occurs if you don't design your machine code routine to do a bunch of different things each time it is called to offset the time wasted in calling it, usually by hidden routines built into BASIC. That is the first thing to remember in writing machine code routines to be called from another language, that is to make sure that they do enough to justify the overhead of calling them. A second thing is that writing compact code is not really necessary in these calls but writing logically simple code that is easy to debug is. So expect your routine to be of medium size at least. Also avoid doing something that requires the calling of several machine code routines to do. The writer wrote a few little machine code routines to replace a FOR ... NEXT loop, one to initialize, and two to handle the beginning and end of the loop. On getting out the stop watch it turned out to run actually a fraction of a second slower than the BASIC's FOR... NEXT loop! That is probably the third commandment of machine code programming. Prove that the machine code is a benefit by testing it with a stopwatch and a test routine that will call it enough times that you can measure a difference in speed of execution with a stopwatch (or use the built-in timing mechanism of your compter if you want a more sophisticated approach).

The second big factor in m/c calls beyond the speed question is that of getting already running routines and data structures like floating point numbers, arrays etc. not interfered with or interfering with the machine code. Another related question is finding good and reliable ways to pass parameters to the machine code and back to BASIC if that is necessary. Another useful thing to remember in BASIC is that many of the things that some amateurs think needs to be done with machine code can instead be done in BASIC. That is why the PEEK and POKE commands are there. But you must know as much or more of the BASIC layout, datatypes, screen file location, etc. for using PEEK and POKE as with the use of machine code, so the average BASIC programmer is not really familiar with the wide range of things that can be done with PEEK and POKE, and in some cases, even faster than in machine language, due to the overhead mentioned above, and that in addition to a machine code call, you may need a second BASIC line to plug the information from the call into a BASIC variable, so it will work with the rest of the program which is in BASIC or course. For example, in the Sinclair QL, being a typical 68000 BASIC in that it takes into account that you may not wish to simply PEEK or POKE byte by byte as in the older, 8-bit computers, you can use PEEK and POKE to access a 16-bit word, or a 32-bit quad byte from RAM. In SuperBASIC on the QL, you can use complicated user programmed functions (similar to subroutines) to do rather complicated functions in BASIC, with PEEKs and POKEs used within the procedure definition that may offer a viable substitute for machine code routines, If you would like to speed them up at a later date , then a machine code subroutine can be written to duplicate their function, once the program is entering its final stage of pre-release optimization, in the case that you want to distribute it to others.

Pairs of 32-bit registers are used in some compilers to handle floating point numbers, and in fact, that was one of the design points favouring the use of 32-bit general registers in such computers as the VAX 11/780. Perhaps some minor use of machine code subroutines with BASIC or other languages, on 32-bit computers could speed up data and number handling in programs. The 32-bits allows signed integers up to two billion (2,000,000,000) so that integer arithmetic itself can be used in applications that would in a computer using a smaller register size need floating point numbers or scaling of integers. Even the compiler or BASIC with the computer may not be fully optimized for use of these wide registers, having merely been adapted in a minor way from a version originally written for 8-bit or 16-bit register computers.

The power of the 68000 is there for you to take advantage of if you have a computer based on it, like the Sinclair QL, Atari ST or Commodore Amiga. It is also a simpler microprocessor than the IBM compatibles since it does not use the complicated segment register/indexing system for addressing RAM in 64K pages, and in general is free of a lot of the clutter of machine code programming conventions associated with 8080 to 8088 microprocessors with their compromises, being in their nature attempts to design microprocessors with many general registers before the technology made that feasible, and then having improved designs tied to some measure of compatibility to the older limited designs, perpetuating the difficulties.

....

How Do I Read That Darn Disk? Compatibility Problems With Other Computers

How do you read the data off a disk used by/written by another computer than the one you own, or the one to get the data off it? You can try sidestepping the whole thing by using a null modem or some other sort of wire to transfer the data rather than exchanging disks at all. Assuming the easiest case, that of text files that you want to read off another, alien disk only, some of the pointers that follow may help, since the problems are more often software than hardware incompatibility.

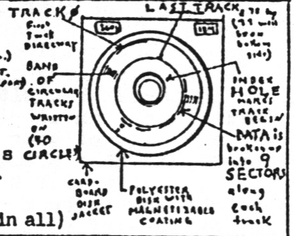
The key to reading data off another disk and the point at which a computer will, when not being driven by some sort of special utility with low level (mainly machine code subroutine) access the alien disk initially is in reading off the info on the directory track which includes both the list of text and program files on the disk, usually in ASCII and a table of how the sectors or blocks on the disk are used, often in a non-obvious way. This serves as the table of contents and index of a book does and it is absolutely essential in quick access to the disk contents. It also is the entry point of the normal disk access routines on your computer and the means for the operating system to reject an alien disk as unreadable with its normal methods.

Where is this directory track? On CP/M floppy disks it is normally the first track, on 320K 5¼ inch IBM PC and 720K 3½ inch IBM PC disks it is normally the first track too, although for the 5¼ inch disks this is an older, MS DOS version 1.25 disk standard for formatting, the industry standard for compatibility on the IBM PC and clones being the 360K 5¼ inch disk in which the directory information also runs over to the second track (track 1 versus zero, on the bottom of the disk). Some computers only use the first 35 track/head positions (sometimes called cylinders) like the Radio Shack Color Computer BASIC disks and the Commodore 64 disks, which are also only single-sided. They also put the directory track in the middle of the bands, rather than as the first track, which is on the outside, track 18 for C-64 and 17 for Coco. CP/M favours the standard way, having the directory track on the first track, as in the older MS DOS standard. Unfortunately CP/M likes to put a track map in RAM and operating systems normally don't do that, resulting in lots of incompatibilities when CP/M utilities etc. that use low level disk access calls are run on non-CP/M computers using a software emulator only. One more note: a simple hack when experimenting in disk transfer if you have access to both types of computers involved in the transfer is to put, using low level or track writing calls, the directory track your computer reading the alien disk wants to see on a disk, on the alien disk, ruining some of the data in the process, but perhaps getting quick access to the disk with a minumum of programming. For using an IBM PC clone to read an alien disk, the 320K 5¼ inch format that rather than the 360K will clobber only one track and may be best.

Fig. #1

MS DOS 5¼ inch 360K DSDD Floppy Disk Specs.
VERSION 2.1 — VERSION 3.1 - VERSION 3.2 - VERSION 3.3

Track 0, top side
sector: 1-boot m/c &
2.3-F.A.T. (ORIG.)
4,5 - EXTRA COPY F.A.T.
6.7.8.9 - DIRECTORY (cont)
Track 1, bottom side
sectors 1,2,3 dir.
Track 1, data sec=4-9
Tracks 2-79=data/files
512 bytes per sector
cluster=2 sectors
9 sectors/track (80 in all)


TRACK 0
First track directory
BAND OF CIRCULAR TRACKS WRITTEN ON (80 CIRCLES)
LAST TRACK, #79 (79 with [illegible] bottom side)
INDEX HOLE MARKS TRACK BEGIN
DATA is broken up into 9 SECTORS along each track
CARD-BOARD DISK JACKET
POLYESTER DISK WITH MAGNETIZABLE COATING

Since the IBM DSDD, 5¼ inch 360K disk is the commonest for non-IBM PC compatible computers to want to read, a summary of its specifications is given in the box above, Fig.#1. Typical contents of the directory track of a disk include as in the IBM PC compatible 360K disk, a boot file, machine code usually with a jump near the beginning to jump over, or simply jump somewhere else, so the next block of info is not executed as machine code and can contain the disk specs, including some info in ASCII easily readable by the hacker using a disk fix/recovery sort of utility. Then there is a list of files on the disk, with some notation usually opposite each one as to the sectors or blocks/tracks where the data of the files is kept on the disk, and a big table of disk sector or block use, called the FAT (File Allocation Table) on the IBM PC, BAT etc. on other computers. Those are the three principal files on a directory track.

Overcoming disk incompatibility is a matter of substituing some form of machine language or low level calls for the normal disk access routines in your computer, since those of your computer normally won't work on an alien disk. File fix/recovery utilities for your computer may have done the hard work already and many computers CP/M and MS DOS especially have full sets of disk conversion utulities out there in commercial, shareware or public domain (usually on BBS systems for the latter).

Disk systems differ in that most use sectors, sections along a track with an identifying header, although some used a whole track at a time. In the IBM and CP/M world 256 or 512 byte sectors were used, a fixed number per track. Of course outer tracks, being circles are larger than inner ones, so some computers like the Commodore 64 use more sectors on the outer than inner tracks although the operating system of that computer has no problem reading a fixed number of sectors (as in C-64, CP/M). IBM PC's take as the smallest snatch of data around a track that can be read in one time — the cluster, which is one or two sectors most often, while most other systems of earlier origin can always read or write to individual sectors due to their DOS program design. Note the existence of hidden files on a disk that do not show up as files on the directory. In a sector system the DOS when it wants to write on a sector, can single out the sector along the track to write on by swooping on the sectors, reading their headings until the wanted one is reached, then switching to read mode (read/write heads are used) and then writing the data to that sector, the operating system allowing some blank bytes on the track for the head to "dribble" random bytes in as it switches from read to write mode electrically. AN AMATEUR PRESS PAGE

- North/Am - 1-92 -

CP/M For All - Including Non-CP/M Computer Owners

Why a column on CP/M in a newsletter not devoted to CP/M computers? Well, the answer is that CP/M can be run on many, many non-CP/M computers by the use of hardware accessories or merely on most by the use of a software emulator. For these computers it offers the closest approach to a standardized disk command set for copying and transfering disk files and some dialects of interpreted and compilable BASIC (mainly CBASIC compiler, MBASIC interpreter and BASCOM compiler) that have been reasonably well documented and save one the problem of coping with eccentric BASIC commands and quirks as you move from one computer to another.

Speaking of which, the answer to a possible future question of many is, "Yes, you can take CP/M with you when you move to an IBM PC compatible in the home or at the office, at least for the most part." The exception is that emulators, especially software ones may not emulate all the CP/M low level calls that certain utilities mainly, need. The solution is to seek those utilities of versions that will run on your computer. About software emulators, most 68000 computers as well as the IBM PC compatibles will run emulators some available even in the public domain. They will keep you going as a BASIC hacker-programmer for as long as you want. Even the Apple Macintosh has a CP/M emulator! And tons of software are available especially games and utilities, from BBS systems that specialize or are well known for CP/M support. Dealing with a BBS for getting your software also solves the problem of incompatible 5¼ inch disk formats that plagues CP/M although even there a BBS may have some programs to convert between different disk formats. If your computer can't normally read an MS DOS 5¼ inch disk and you have a 5¼ inch drive, you may be able to find a version of an MS DOS disk reader and even text-file writer, in a CP/M utility compatible with your CP/M emulator. Even though it won't enable you to run MS DOS programs, it will let you see what is on a disk and read text files most likely.

Other than the disk compatibility problem there is the screen compatibility problem, the easiest to visualize being the problem of different computers, even including a couple of CP/M ones, using different widths of screen as measured by the number of characters maximum to a line across, called the columns of the screen. This means that text may scroll off or be hard to edit with a word processor. The usual standard for CP/M is 80-column although computers using emulators may be only designed for 64, 40 or 32 columns to mention some of the commonest variations. The ultimate solution by way of a quick fix at least would be to write a custom display or text edit routine or find such a utility tailored to the screen size of your own computer, the one running CP/M under emulation. Since memory is at a premium in CP/M computers this may mean an unexpectedly complicated program is needed to edit text, one with either very small demands on memory, and so needing to be a compiled program, preferably in machine language to save space or the use of complicated routines to lay-off text in temporary files on disk. Another problem is that some computers or the versions of them made on some CP/M computers need a different device driver for printing on the screen or paper, although this is not something that one might expect to encounter with most CP/M software emulators or CP/M user programs, like utilities or word processors. But it does happen.

In passing, let us just mention that using CP/M is possible on the IBM PC or AT compatible thanks to a number of software emulators, more than one, like Z80MU being in the public domain. Also the VMU emulator can be used if your IBM compatible has a V-20 (or V-30) chip in place of the 8088 or 8086 (respectively). The V-20 processes both 8088 instructions and emulates 8080 instruction processing when switched over to 8080 emulation mode with a machine language instruction unique to the V-20. Since the V-20 (and V-30) use less power, being CMOS chips, if you are using a laptop, it may already have one of these chips installed, if that is, it is one of the older models of laptop that does not use an 80286 or 80386 microprocessor. The VMU emulator for such computers does run a little faster, but still better speed is available with the non-public domain emulators that the same company sells. Even with the public domain Z80MU however, an IBM PC or AT with an Norton SI speed rating of about 3 to 4 should run CP/M programs on the emulator about as fast as a 1 Mhz. CP/M computer like the early ones or an Apple II running CP/M with a Z-80 hardware/emulation board. With so many turbo models of the IBM PC/XT made, even a non-80286 model with only a speeded up clock should run CP/M as fast as you will need for ordinary work, using only the Z80MU available free in public domain form from the BBS's or mailed from a shareware disk vendor for about $7, typical (CP/M programs and related products do often command a premium price when going through commercial hands).

Office Automation: The Second Industrial Revolution?

Yes we did survive the industrial revolution and our democracy became even more democratic at the same time. But will the middle class survive this second revolution of technology, the automation of information used to control economic activities at every level of production, factory robots, retail inventory control and sales and the automation of most office jobs as well? Let's start with the examination of the middle class. It is literally the class between the rich whose comfort and security is of a high level and the lower class, largely employed if at all only intermittently to which we add the working class, employed in one, steady job, but on an hourly basis. Many jobs that used to be considered working class because of the hourly fluctuations of emplyment and low wage levels became lumped in with the middle class as fluctuations in working became less and wages higher in the prosperity of this century. Many other jobs as well as part of the working class sort of manufacturing jobs, also became part of the middle class as the need to function on a level of literacy and number handling, mainly accounting, in so many jobs, started to merge with that necessary in middle class business/office jobs. In this century the pay for such jobs and the number of such jobs vastly increased, making the prosperity of the twentieth century seen to be based on this expansion of our middle class. Then came the computer revolution, in which the craft-like, customized handling of numbers and words that this middle class prosperity was based on is being done by mass production and by another machine, the computer, linked in networks, so that not even an office boy, much less a whole level of instruction and order carrying and carrying out managers, need trot between nodes in the network, factory/office with a bundle of papers under their arm. The resultant shrinking of the middle class has led to a spiral downward of the pay of those remaining in it, as larger numbers compete for fewer jobs. An article in Scientific American magazine, of Feb. 1992, p.111, documents the exodus of numbers from the middle class, mostly down, not up.

Personally, this writer disbelieves that we are seeing the effect of a post-industrial society* where fewer are needed to manufacture goods, and more needed to create or process information. That process has only slowly started to shift into effect and instead, the contracting out of labour-demanding manufacturing to low wage countries has preceded it. The fact that jobs may be lost due to increases in efficiency and pay money lost then also being temporary, may have been true in a time that loss of manual manufacturing jobs was replaced by craft-like and manual work of information processing, but in this cycle, it is the middle-class information handling jobs that are disappearing at the same time. They may be replaced by other jobs or they may not, or the jobs replacing them may be more poorer paid than better paid in this economic phase, the opposite of the situation in the last one. If the size and prosperity of this information handling middle class has been reduced so much, and this middle class and its money has fueled a demand for a certain quality and quantity of consumer good, then the consumer goods that the retail, etc. sector has been furnishing may no longer find a market, or certainly not the same or as big a market. Similarly, those oriental companies and governments that have invested in manufacturing plants on the assumption that the capital debt could be paid off by the same level of demand and high priced product that the purchases of this once larger and more prosperous American middle class used to make, may have a lot of bad debts on their hands in the future. A low wage manufacturing country used to gradually become a higher wage country, but this pattern may not be repeated tomorrow. It is obviously a very big change in economic structure we are seeing here, not a blip.

At the grass roots level, where you and I live, what can this mean? For one thing if the deindustrialization has not yet really hit (or may never), this alone will put a different answer to the question of where the future jobs will be found. Personal service is likely to be the buzz word for the lower level jobs, restaurant, assisting, helping make choices and customize purchases and services, right down to social, etc. secretaries and housekeeping. On the higher level, it will be the age of the consultant if it is not already. This certainly means that technical and professional people will need a broader range of skills to be in this line of work, for like a farmer, a consultant will have to keep the books, fix the office chair and computer connections, type the bills and calculate the amounts, sell the services in the first place and do the promotion and public relations and complaint adjustment. We are not talking of the generalist instead of the specialist, but the generalist skills needed to be added to those of the specialist. One cannot know too many different things today.

Finally, jobs will be found that do not exit today, just as word processor and CAD and desktop publishing jobs did not exist in any number, just a dozen years ago.

Possible New Directions For Processor Hardware Designs

It is hardly surprising that CPU or microprocessor hardware should be designed in order to do fastest and in hardware versus software emulation, what it is going to do most. The least frequent applications can be accomodated by emulation, where several software instructions in machine language do what another, more efficient CPU or microprocessor might take only one short m/c instruction to accomplish. In fact, if you look at the brief history of microprocessors, you can see that they have already been changed in order to optimize them for new tasks, the 4004 by Intel, the world's first microprocessor, having 4-bits, suitable for BCD arithmetic of small office calculators, being expanded in the 8080 to handle the 8-bits of an ASCII character (plus one bit for parity checking) when it was seen that microprocessors could radically decrease the number of parts and so the cost to manufacture, dumb video terminals to connect to the big computers of the time, in time-sharing installations. The 8080 also had certain registers 16-bit wide, that could hold the whole address of a 64K memory map (ROM and RAM) location, so that strings could be handled by the processor by INCrementing the originally stored RAM address at the start of the string so that the microprocessor could work its way along the string, one character ahead at a time, moving, checking, comparing or modifying each byte-sized slot for an ASCII letter code, as it moved. (Further History-the 6800 and 6502 were made as controllers rather than computer chips, and the 8080 and its immediate and extinct predecessor the 8008 were designed for terminals rather than self-standing computers although they were soon used for them.)

The current generation of 16-bit microprocessors and their 32-bit upgrades are descended from two lines of parentage, the 6800 8-bit controller, with two general registers on which all of its data instructions worked, and the 8-bit 8080 which had a variety of special-use registers in addition to one accumulator as the general register. 8080 special-use registers often only allow a few of the commands that you would expect a register to have, such as INCrement (add one to contents) and DECrement, (subtract one from contents), and instructions to load then with a specific number in the program code itself, or get a number to put into them from another register. These are quite differnt microprocessor architecture philosophies, the 68000 etc. descending from the 6800, and the 8088/8086/80286/80386 descending from the 8080. A m/c programmer moving from working from one must really shift gears as to what it is possible to do with the registers of each family, since the same thing may not be possible on a microprocessor from the other family, although the distinctions have been lost as each evolves towards the middle ground of similarity to each other. For example in the 8080 you cannot* add 5 to certain registers, just INC them 5 times, and **INX**/INC BC may not set the right status flag when the job is done, as a 6800 would, with for example its registers. (To fit in the Z-80, it is descended from the 8080 and as ~~the~~ 6502 from the 6800, both being 8-bit, slightly enhanced or changed versions, produced by companies competing with the original makers.)

What about other computers than microcomputers? Well, earlier computers when they were adapted from scientific and engineering number crunchers, moved away from the chunk of binary bits that they handled at one time being based on the largest octal digited number that additional hardware to handle would not be too expensive to add. So computers that had BCD numbers (4-bits) yet for which two of them could form the codes for letters and punctuation, needed in business data processing, etc. like the IBM 1620 and IBM1401, became the paradigm, massproduced to make them available to medium-sized government and larger business operations. That was an early compromise design to keep number crunching ability yet add efficient text handling. This hybridization of string handling and number crunching in processor design put IBM ahead of its earlier, and technologically superior competitors that continued to specialize in the smaller market for expensive number crunchers (Control Data, being a good example). It has seemed to work well in microprocessors, with the 8-bit designs, the 8080 text-handling-terminal-CPU and the 8-bit controller 6800, evolved into the 8088/8086/80286/80386 and 68000 respecitively, by adding more and bigger registers and eventually math-coprocessors to handle decimal fraction math, which is much more processor-demanding than arithmetic with integers (whole numbers). Now the number crunching Cray computers and HP, etc. RISC computers have optimized for number crunching by being able to handle better fractional floating point numbers and vectors, those lists of numbers, where one m/c instruction can operate separately on all the numbers on two lists of numbers, while an ordinary (non-vector) CPU or co-does this on two numbers themselves, at one step. (Imagine two whole columns of numbers in a business spreadsheet being added in one millionth of a second.)

⟶ continued next pg.

*or any number other than +1 or −1, (INC & DEC).

Since much scientific and engineering even, number crunching is done with vectors (vector equations) these days, the big number crunching computers are being tailored to handle such vectors faster and at a more basic machine code and hardware level. Why machine code and hardware? It runs faster if the hardware is better structured so that it ressembles the structure of the number it is handling. It is like using a dozen cars to drive the football team to the next town for a game as opposed to using one bus. Except using a dozen cars would be parallel processing with a dozen processors, while the normal, non-vector computer would use one car and take 12 separate trips. So that is why if you design the processor to do in one operation 12 things or handle data with 12 times the size that a normal processor could handle it will run 12 times as fast, and with less programming (at the m/c level). Doing more in hardware is inherently faster* and doing more with just one instruction, means fewer instructions need be used and so fewer trips to fetch them from where the computer program is stored, operations that tie up the bus, which, being made up of longer conductors, presents lots of technical problems to the engineer trying to speed it up. (A bus tends to be electrically spongy rather than firm and crisp, sort of like trying to hit a fly with a swatter attached to the end of a fishing rod instead of a swatter with a shorter handle.) More of the number crunching computers are also dealing with the problem of reducing the amount of traffic that must travel on a bus, by using the so-called Harvard architecture, using two busses, a form of parallelism, one for instructions (m/c) only and the other only for data. E.g., the Motorola 88000 RISC microprocessor systems. Using two busses like that doubles speed.

I do not know how you use your computer. I use mine for wordprocessing and practically no CAD/CAM, 'scientific' calculations or design. So my computer does text handling nearly all the time and most of that needs only a microprocessor that can INCrement and DECrement along RAM addresses where lines of 8-bit chunks of text are stored, the writing that I am working on at the time. Does this mean that word-processing uses no arithmetic beyond that? Having tried to write some word processor computer programs myself, I can guaranty that you do need arithmetic, even multiplication, division, and other operations, to decide how many lines go on a filled screen when you ask the computer program to skip from one page to the next, and so on, although this is both integer math (whole numbers, no decimal points needed), and fairly simple. So text handling and arithmetic are needed. Of course, a register with 16-bit capacity will flip though your text two characters at a time in a text move or output, rather than the slower, one-character at a time of an 8-bit microprocessor, so an IBM AT should be faster in at least input/output transfers or block moves, than a Z-80 based Osborne, if the software takes advantage of the IBM AT's wider registers for these operations. CP/M computers and the 8-bit data bus, IBM PC also at one time tried to speed up data transfer with a sort of extra piece of hardware, a Direct Memory Access controller, that would do fast block transfers, when disk copying, as one practical example, rather than the way that the microprocessor itself would do it. Direct Memory Access can also run in parallel to the microprocessor doing something else, (at least part of the time), so that some of the work is being done when the microprocessor leaves the bus free, the time saved being a complete bonus which increases the speed of such a block transfer. Minimal mathematics, integer mathematics only, speed in block transfers by designing hardware like a DMA, ability to INcrement and DECrement though long stretches of 8-bit text over the whole or a large part of the RAM computer memory; these are the main features to optimize in a microprocessor and related chips, for a text processor computer of the old type, the CP/M, MS DOS, Z-80, 8088, character based, dumb terminal family-tree descended, and so on.

But there is something new under the sun in home computers. It is the graphics based screen. It shows things in pictures, windows open up one over the other in layers in the attempt to keep different tasks being run in different programs sold by different companies, all working in the same, (one) computer, and the screen used to keep these in view when needed. This is the world of the Graphics User Interface, of the Apple Mac, The NeXT cube, the GEM operating system extension and the Microsoft Windows (TM) for MS DOS computers. It has added some new requirements for efficiency and therefore optimization of processors at the m/c, hardware level, both for microprocessors and accessory processing chips. These computers must be very efficient in handling not 8-bit character codes, and addresses for 64K memory, but much larger RAM memory addresses than the old 16-bit register that held it for 64K of RAM, can handle. Text is handled for screen display in long 'frames' of bits, a binary bit for each dot on the screen to slightly oversimplify, since colour is

CONTINUED NEXT PG.→

* software emulation usually uses more than one m/c instruction to emulate a system's one instr. or electronic operation in hardware (emulation or native mode).

New Directions In Processor Hardware Continued

usually handled as well and separate from the 0 for a dark point in a line of dots on the screen and 1 for a bright point, (the so-called bit mapping of graphics). These frames* of dots that would draw short lines on the screen have to be operated on at a very low hardware level to make handling them fast, when one line overlays and then modifies, part of another, (say as one window opens up on top of part of something else). So two lines are going to overlap. What to do with the spot where they overlap? You might perform a logical operation on the dots, bit by binary bit, and you shall need it done fast, which means hardware, which means a processor either inside a new, restructured microprocessor to be invented for this, or a special purpose silicon chip processor to do this work outside the microprocessor. A new structure of microprocessor for a new structure of data, a frame of dots or several that need to be made into one line of dots, overlaps and putting the resultant frames to be displayed, end on end until they form a complete scanning line for the screen, as the CRT's electron beam paints a pattern of dots on the monitor screen in a line from left to right, at a speed of ~8,000 mph in even a slow, crude tv used as your monitor. Obviously it is a matter of time until specialized coprocessors or even completely redesigned and graphics-optimized microprocessors are on the market to handle bit/line graphics. That is of course only part of it, as for the calculations needed to do speedy graphics, especially three dimensional which is coming in the future, even in desktop/windowed programs, you need some vector-processing, since vectors are useful in speeding analytic/coordinate geometry to handle these graphics. (Unlike supercomputer vector processing, perhaps integer rather than floating point vectors would be enough, simplifying the processor design complexity.) Then a third type of optimization is needed for microprocessors and other processing hardware for future graphic oriented computers, and that is the sort of hardware that allows crash-free processing of different programs in the machine code level, in other words multi-tasking. For crash resistant multi-tasking, you need to hermetically seal off those operations that would, if two programs tried to access them at the same time, result in crashing the computer (having it lose its place and start processing garbage as real instructions, etc.). The popular way of doing this is to provide a protected mode (like the 80286, 80386, 68020 and higher), so that one program, the operating system will be the one chosen, has access to the main control of the microprocessor (called sometimes protected mode or kernel or supervisor mode in other computers) and it only can run certain instructions. This will avoid crashes caused by the left hand not knowing what the right hand is doing, a good thing when the right hand is making an anonymous contribution to a charity but not when one program in a multi-taking computer is scrambling things that a second program will rely on being a certain place and form when it comes back to displace the first program and take its turn of running on the microprocessor. So to design an ideal computer to run the new Graphics User Interface software and enhance its features, you would need to optimize the CPU for 3-D vector math, bit-line or 'frame'* processing for windows, and multi-tasking capable microprocessors that do not need the artificial software fixes to multi-task that the Z-80, 8088, 8086 and 68000 need. At the moment the use of extra processing chips to do this has been seen in the Commodore Amiga, so that frame* processing (a part of a whole line of bits) is done in hardware although the multitasking is done with no hardware facility like a protected mode since the Amiga's 68000 has none, and no fancy vector-coprocessor chip at all. But in the Amiga we can at least see the shape of future hardware optimized for the GUI age of software.

But is that all? Not exactly! The priority in business computers is for networking, since in business and government, microcomputers in networks once used in a dual capacity, as stand-alone wordprocessors and data entry/processing computers and as terminals for a larger, more costly minicomputer or IBM mainframe, now are being used without the big computer, and connected with software and networking cable so that they do not need any big computer anymore to be connected in a network. Networking software is expanding from the point that a printer upstairs or a bigger than average file-server microcomputer downstairs can be used by any of the microcomputers connected in the network, to an even more advanced function. Networks are starting to work in a way that a job that would take too long on one microcomputer can have its processing job split up so that other computers and on other floors can help out by doing some of the processing. This may be done is some systems, even if the helping computers are all being used for something else at the time, but just not to their full capacity, the extra being made available to the computer on the other floor that otherwise would be overworked by a big job. At the moment extra network boards are being plugged into such computers but one day processors will be redesigned

(*frame = Amiga designer's term for a scan line segment less than the full screen width.)

Don't Count Magnetic Media Out Yet

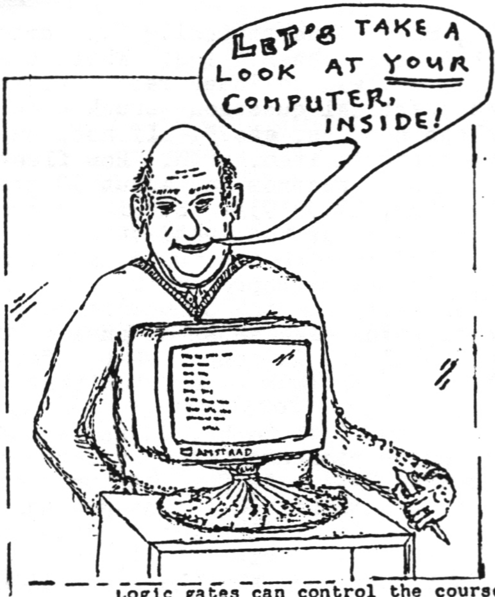
In the market place's commercial battles, many a theoretically fine product has gone to its grave, sometimes before it has really been tried. While standing in the second hand shop in front of an old Edison wax cylinder record player, thinking perhaps, dreaming of buying it, the practical question struck home, "Can one still find wax cylinders, plain steel phono needles, etc?". If not, such a device would be more a dustcatcher than a collector's item, even. How fleeting is the life of a musical medium! The large, long play records had about 30 years, 1955-1985, the 78 rpm (smashables) about the same, 1925-1955. But in my granddad's day, the musical toy was one with digital input and was the quasi-automaton of the 1920's, the player piano. Whether future ways of reproducing music and even vocals will go a similar route to the player piano, via the microprocessor, would make a very risky wager. Nonetheless the newer home computers like the Commodore Amiga and NeXT cube have very elabourate music features, which make those of an IBM PC seem like the spoons and washboard of the hillbilly culture in comparison! Perhaps it wouldn't be via a full featured computer at all, but a'music computer'that would cost as little as a CD disk player does today, for 'robot' music making.

But if I were a betting person, I would be inclined to bet on semi-conductor memories (like the Sinclair Z-88 notebook computer pioneered from as early as a few years ago), and magnetic media if new formats and kinds. As for the music market, while one has to admit that CD disks have not been the flash in the pan that the video disks were, at the very least, a smaller format is likely to become popular for computer use as CD ROM (and maybe WORM or truly erasable disks), and maybe too, music fans will find the smaller CD format just too handy to make them stick to the old format. The present CD disks have two big disadvantages for computer users: they are too big for laptop or even convenient desk top insertion/handling and you can't record on them. If you can't record on them, after you have a dictionary, thesaurus and encyclopedia on disk, what else in the way of mass market stuff can you need? Mind you, if a lot of specialty publications go on to CD ROM like the Physician's Desk Reference, 'Rubber bible', integrated circuit master indexes etc., you may see a greater scope of use for them. Once the read only feature and the size limitation (and already a smaller size CD is on the market) are overcome, the CD ROM/erasable disk, could take off. But would it be temporary? Do we see 30 years ahead for the CD ROM in its present form or even in a variety of mutated sizes and types (each requiring a new generation of player or drive)?

Well, I for one, would not discount a renewed lease on life for the magnetic disk. The 2 inch disk may soon become the standard for low-density disks, just as the 5¼ inch disk starts to seem rather large for its 360-400K of data and the 3½ inch disk may never really establish itself. The 8 inch computer disk reigned from 1973 to 1983, replacing in mainframe computers, the IBM 350 disk drive units of 1957 origin, to a large degree. The 5¼ inch disk was born in 1976 and the 3½,1981?. But now we have gone from single-density (100K maximum per side) on the 5¼ inch, to double-sided single-density at 200K (or single-sided double-density, the same 200K), to double-sided, double-density, 400K max, to high-density with double the number of narrower tracks as well, at 1.6 megabyte max (1.2 meg commonly). Can we have super density as the next level of improvement at four times that or merely double? With the humble 5¼ inch disk, that starts (with archived data compressed with say the ZIP utility) to bring us up to the 10 megabyte capacities of the first generation of built-in disk, hard disks (then called Winchester disks, but no longer referred to as anything other than hard disks).

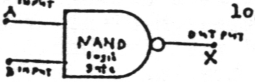
Those who are saying that magnetic disks and media will soon be obsolete as the laser disk/optical disk form of the CD ROM and musical CD disk take over are going to be proved wrong in the estimation of this scribe and amateur crystal ball futurist. (Track record so far: predicting the early demise of the video disk in favour to the thriving of the magnetic tape based VCR.) Neither should you discount the digital or improved fidelity analogue magnetic tape medium for music as opposed to the CD disk, at least in its present bulky, non-recordable format. Thirty years from now, the browser in the second hand shop will spy out a CD disk player and ask quietly how it would be of any use except as a dustcatcher, since CD disks are rarely seen outside museums . . . .

A Further Look Inside Computers For The Electronics Student


LET'S TAKE A LOOK AT YOUR COMPUTER, INSIDE!
AMSTRAD

A digital computer is made up of many small circuit building blocks, arranged exactly to make a computer. The Boolean logic logic gate is one of the most basic. It can process binary numbers (0 or 1 allowed only) to do arithmetic or the true and false of logic.

Fig.#1



This NAND gate can be used to build a whole computer. But it itself is made up of smaller elements, in an electronic computer transistors inside silicon chips. Here next is the exact circuit used for a NAND gate at one time, to do what Fig.#1 shows in logic symbols.

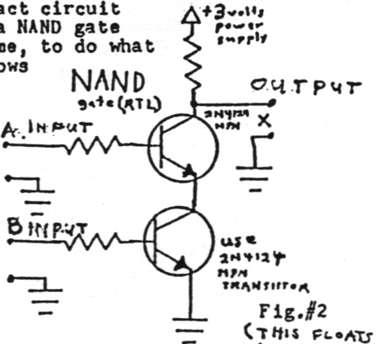


Fig.#2
(THIS FLOATS INPUTS LOW VS 74LS00 HIGH)

Logic gates can control the course of calculating or moving data in a computer according to certain conditions fed into the input of a logic gate, a High used as True for example, and a Low (near ground voltage) as saying the condition in question is False. (An example of a condition it may be false "that a number stored in a storage spot like a register is zero." The computer may have to know that, in order to decide if the calculation needs one more step or is finished.)

These logic gates are very useful for they can do arithmetic also. The Fig.#3 circuit shows several NAND gates hooked up together to add two, 1bit binary numbers. So Boolean logic and arithmetic can both be run with the same NAND logic gates. So they are simply used for that too inside computers. A third job is to rewire the circuits in the computer to do a different job, something the computer does in the microprocessor or CPU for each new machine language instruction. To save the labour of hand rewiring as done in the old days, the logic gates themselves can rewire or at least make different connections between the wires. In Fig.#4, right, putting a high to input S, will connect data line D to wire B, instead of A. A lot of these circuits can make a computer programmable, the numbers doing the programming called the machine language instructions or simply machine code.

Fig#3 circuit to add two binary one-bit numbers a_1, a_2

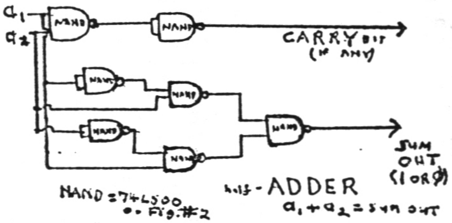


Fig.#3, NAND gates used for adding, binary arithmetic, another important use of the logic gate as a building block for digital computers—from this simplest binary adder circuit, much more complicated ones are built to add more than one binary digit at a time

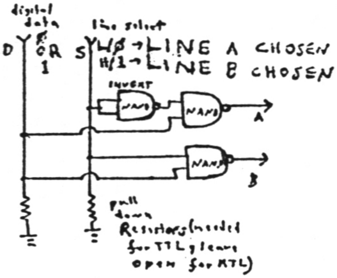


Fig.#4 - line select
(deMUX in tech terms)
NAND = 74LS00 or Fig.#2

One more thing that logic gates can be made to do, by building bigger, special purpose circuits of them—to store a number as in RAM memory or fast registers in the microprocessor or CPU. In this NAND gates are connected to form a flip-flop, which will store one binary digit but many flip-flops together can store big numbers, as big as needed.

—The ability to build so many complex electronic circuits with such small parts means the simplest computer will have about 300,000 transistors used to make these circuits today. (Note cost requires NOT and NOR used too.)

B.J.H.—copyright ©June,1992. in the public domain
—copy freely!—

What Is A Computer? Searching For The Ultimate Answer

If you have ever been asked by a young child what a computer is and then analysed your answer afterwards, you likely have found it to be seriously lacking to your own ways of thinking. What is a computer? Turn to the dictionary and you likely will only find a definition that will satisfy someone who doesn't know any better. It talks most often about high speed, processing of data (and who doesn't from ant to architect, fool to philosopher, so that hardly makes any special claim), arithmetic and logic. So arithmetic only distinguishes it from animals not as clever as "Clever Hans" the circus horse who could tap out answers involving addition. Maybe the heart of such definition is that the computer does only those functions of people or animals. Then call the control circuits hooking a computer to a robot or the engine in your car, something different than being part of the computer itself, as you define computer and you might have the start of a real definition.

Kids are much easier to handle and most adult learners; you just sit them down in front of a computer and show them what they can do with it. Computer is as computer does. Today most of this involves using the computer as a word processor, a way of souping up the performance of an electronic typewriter to the point that even a non-typist can use one. So when today's computer user thinks of a computer they think in terms of graphics, boxes, colours, and as well things like word/character handling, and do not see at all the original definition of a computer as something that computes. We as amateur programmers see it however. Everything that you ask a computer to do is done by computation, rather than estimating: how long a line should be to balance the graphics or line lengths of your essay or personal letter on the screen, and how many lines to display. All calculation. In short it doesn't think the way we do, it just calculates what it is told to, not knowing or particularly caring about the consequences in an individual case. It is figuring blind. The programmer does the caring in advance, trying desperately to figure out the consequences of the particular formula chosen, among other possibilities, to make sure that it will never lead to an unwanted, careless seeming result, no matter what case, even highly unanticipated, that the formula will be used in, say to get all the line lengths to fit on the screen.

So when an article came out recently in the media that talked of a computer not being a calculator in that nowadays it mainly did graphics and calling it a computer as being a word implying calculation (but being a little different, implying more complicated mathematics perhaps) belonging to ancient history of computers, I got mad. It simply is not true. It may not seem to be calculating but inside its intelligence is based on calculation, and often, depending on the application, no more except data handling and a few other details seen only internally.

What is a computer? It is many things to many people. To programmers like many of us it is certainly a calculating device. To hardware people it is something else, additionally, but also illustrative to look at. For that reason, a brief perspective of the electronic nature of computers and how they actually do the calculation and program running and data moving, is placed on the page opposite this. Take a look at it if you are curious. But don't spread the malicious rumour that computers don't do any calculation any more. Technical people will think you are a bit silly if you do. And computers don't need to get the same reputation as high school and college graduates of not being able to do mathematics any more even though previous generations did and even seemed to enjoy it. Computers of all things certainly aren't innumerate, even in today's world. TS Hobby Newsletter-B.J.Harmer

……………………

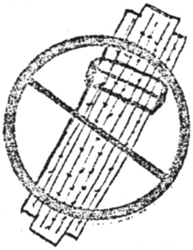
Book Review: Data Structures I and II in the Essentials® of Outline Notes

If your have learned and practiced a fair bit of programming on your computer or are interested in a career in computer science and have taken some programming courses in school, you may be ready to take one further step in deepening your understanding of computer programming. These two books are an excellent way of stepping into the world of real computer science concepts. They are well written, brief and to the point and these thin paperbacks sell for about $5-10, from Research and Education Association, 61 Ethel Rd.W, Piscataway,NJ,USA 08854.

The Data Structures I, covers in 99 pages arrays, sorting and searching as well as linked lists and stacks. Data Structures II goes on to cover binary trees and digraphs, hash coding and advanced sorting in the space of its 200 pages. The writer has never seen some of these concepts explained so clearly elsewhere. They are not beginners books though. But if you are starting to read books on computer software or magazine articles, in which you run into words you do not understand, these books may be the place to look them up and get a full explanation of the whole concept, something a dictionary style book is too brief to offer. Note that they also have books on BASIC, COBOL, Pascal, FORTRAN, PL/1 and math and science.

……………………

INTERNATIONAL Computer SIGNS



ELECTRONIC
COMPUTERS
ONLY

NO SPAGHETTI CODE

DON'T PANIC!
(DEADLINES ARE
MADE AS GUIDES
NOT LAWS)

© 1992, SEPT.,
IN THE PUBLIC DOMAIN

. . . .

In These Days Of Modems, Its not what you knows but what's your nodes, that counts.